What is claimed is:

1. An apparatus comprising:

a register stack engine to trigger memory operations in support of register windows;

the register stack engine further to generate one or more micro-operations to perform a register window operation.

2. The apparatus of claim 1, wherein:

the register stack engine is further to insert the one or more micro-operations into an execution pipeline.

3. The apparatus of claim 1, wherein:

the register window operation is a spill operation.

4. The apparatus of claim 3, wherein:

the one or more micro-operations include a store micro-operation.

5. The apparatus of claim 1, wherein:

the register window operation is a fill operation.

6. The apparatus of claim 5, wherein the one or more micro-operations include a load micro-operation.

7. The apparatus of claim 1, wherein:

the register stack engine is further to generate the micro-operations indirectly, via a micro-op generator.

8. The apparatus of claim 2, wherein:

the register stack engine is further to insert the one or more micro-operations into the execution pipeline indirectly, via a micro-op generator.

9. The apparatus of claim 2, further comprising:

a micro-operation queue;

wherein inserting the one or more micro-operations into the execution pipeline further comprises inserting the micro-operations into the micro-operation queue.

10. The apparatus of claim 1, wherein:

the register window operation is associated with an implicit operand; and

the one or more micro-operations includes a micro-operation that indicates the implicit operand as an explicit operand.

11. The apparatus of claim 10, wherein:

the implicit operand is a status bit collection register.

12. The apparatus of claim 10, wherein:

the implicit operand is a store pointer register.

13. The apparatus of claim 10, wherein:

the implicit operand is a load pointer register.

14. The apparatus of claim 1, further comprising:

a scheduler to schedule the micro-operations for execution;

wherein the scheduler is to concurrently consider the register window operation micro-operations as well as other micro-operations in an out-of-order scheduling scheme.

15. The apparatus of claim 1, wherein:

each of the micro-operations is of a format that includes a single explicit destination operand and two explicit source operands.

16. A system comprising:

a memory to store an instruction, the memory including a backing store to store one or more spilled values; and

a processor coupled to the memory;

wherein the processor includes a register stack engine to generate, responsive to the instruction, one or more micro-operations to cause a register stack operation.

17. The system of claim 16, wherein:

the memory is a DRAM.

18. The system of claim 16, wherein:

the processor further includes an architectural renamer to rename registers to support register windowing.

19. The system of claim 16, wherein:

the processor further includes an out-of-order rename unit to map logical registers to physical registers in order to increase parallelism.

20. The system of claim 16, wherein:

the register stack operation is a spill operation.

21. The system of claim 16, wherein:

the register stack operation is a fill operation.

22. The system of claim 16, wherein:

the processor further includes a scheduler to perform out-of-order scheduling for a set of micro-operations, wherein the set of micro-operations includes a regular micro-operation and also includes the one or more micro-operations to cause a register stack operation.

23. The system of claim 22, wherein:

the scheduler considers the set of micro-operations for out-of-order scheduling such that the regular micro-operation and the one or more micro-operations are scheduled in an intermingled fashion.

24. A method comprising:

performing an architectural rename stage for an instruction, in order to support register windowing; and

performing an out-of-order rename stage for each of the one or more micro-operations.

25. The method of claim 24 wherein:

the instruction is a procedure call instruction to invoke a new procedure; and performing an architectural rename stage further comprises renaming physical register operands for a current procedure such that output registers for the current procedure are identified as input registers for the new procedure .

26. The method of claim 24 wherein:

performing an architectural rename stage further comprises renaming a first input register to a predetermined physical register number.

27. The method of claim 24, further comprising:

generating one or more micro-operations to implement the instruction.

28. The method of claim 27 wherein:

generating one or more micro-operations further comprises generating a micro-op to perform a desired memory operation.

29. The method of claim 27 wherein:

generating one or more micro-operations further comprises generating a micro-op to perform an arithmetic operation associated with a register stack engine ("RSE") operation.

30. The method of claim 27 wherein:

generating one or more micro-operations further comprises generating a micro-op to perform a bit manipulation operation associated with a register stack engine ("RSE") operation.

31. The method of claim 24 wherein:

performing an out-of-order rename stage further comprises mapping an architectural register to a physical rename register in order to minimize data dependencies.

32. A method, comprising:

generating one or more micro-operations to perform a RSE operation; and

inserting the one or more micro-operations into an execution pipeline;

wherein the RSE operation is to support register windowing.

33. The method of claim 32, wherein:

the RSE operation is a spill operation ; and

generating one or more micro-operations further comprises generating a store micro-operation.

34. The method of claim 33, wherein:

generating a store micro-operation further comprises generating a store micro-operation to store data associated with the spill operation to a backing store in a memory.

35. The method of claim 32, wherein:

generating one or more micro-operations further comprises generating a micro-operation to operate on an implicit operand.

36. The method of claim 35, wherein:

generating one or more micro-operations further comprises generating a micro-operation to perform an arithmetic operation on an implicit operand.

37. The method of claim 35, wherein:

generating one or more micro-operations further comprises generating a micro-operation to perform a bit-manipulation operation on an implicit operand.

38. The method of claim 35, wherein:

the implicit operand is a status bit collection register.

39. The method of claim 35, wherein:

generating a micro-operation to operate on an implicit operand further comprises generating a micro-operation to collect a status bit into the implicit operand.

40. The method of claim 35, wherein:

generating a micro-operation to operate on an implicit operand further comprises generating a micro-operation to restore a status bit value from the implicit operand.

41. The method of claim 32, wherein:

the RSE operation is a fill operation; and

generating one or more micro-operations further comprises generating a load micro-operation.

42. The method of claim 41, wherein:

generating a load micro-operation further comprises generating a load micro-operation to load data associated with the fill operation from a backing store in a memory into a register.

43. The method of claim 32, wherein:

the RSE operation is a spill operation; and

generating one or more micro-operations further comprises generating a micro-operation to assign data associated with the spill operation to one half of a double-wide data register.

44. The method of claim 43, further comprising:

generating one or more micro-operations to store the contents of the double-wide data register to a backing store.

45. The method of claim 43, wherein generating one or more micro-operations further comprises:

determining whether a pre-determined number of prior spill operations has been performed;

if not, generating a micro-operation to assign general register data to the one half of a double-wide data register value; and

otherwise, generating a micro-operation to assign status data to the one half of the double-wide data register.

46. The method of claim 45, further comprising:

if the pre-determined number of prior spill operations has not been performed, generating a micro-operation to merge a status bit into a status collection variable.

47. The method of claim 45, further comprising:

generating one or more additional micro-operations to perform a second spill operation;

wherein generating the one or more additional micro-operations includes:

generating a micro-operation to assign general register data to the other half of the double-wide data register; and

generating a micro-operation to store the double-wide data register value to a backing store.

48. The method of claim 47, wherein generating one or more additional micro-operations further comprises:

generating the micro-operation to assign general register data to the other half of the double-wide data register only if a predetermined number of prior spill operations has occurred;

otherwise, generating a micro-operation to assign status data to the other half of the double-wide data register.

49. The method of claim 32, wherein:

the RSE operation is a fill operation; and

generating one or more micro-operations further comprises generating a micro-operation to obtain a double-wide data value from a backing store.

50. The method of claim 49, further comprising:

generating one or more micro-operations to assign one half of the double-wide data value to a general register.

51. The method of claim 49, further comprising:

generating one or more micro-operations to assign one half of the double-wide data value to a status bit collection register.

52. The method of claim 49, wherein generating one or more micro-operations further comprises:

determining whether a pre-determined number of prior fill operations has been performed;

if not, generating a micro-operation to assign one half of the double-wide data register value to a general register; and

otherwise, generating a micro-operation to assign one half of the double-wide data register value to a status collection register.

53. The method of claim 52, further comprising:

if the pre-determined number of prior fill operations has not been performed, generating a micro-operation to extract a status bit from a status collection register.

54. The method of claim 52, further comprising:

generating one or more additional micro-operations to perform a second fill operation;

wherein generating the one or more additional micro-operations includes:

generating a micro-operation to assign the other half of the double-wide data register data to a general register.

55. The method of claim 54, wherein generating one or more additional micro-operations further comprises:

generating the micro-operation to assign a general register to the other half of the double-wide data register value only if a predetermined number of prior fill operations has occurred;

otherwise, generating a micro-operation to assign the other half of the double-wide data register to a status collection register.

56. A method comprising:

generating micro-operations to perform, in a single cycle, M parallel memory operations in support of register windowing, where M > 1;

wherein generating micro-operations further comprises:

utilizing a first memory pointer register to determine the memory address for a first memory operation; and

utilizing a second memory pointer register to determine the memory address for a second memory operation.

57. The method of claim 56, wherein generating micro-operations further comprises:

utilizing an Nth memory pointer register to determine the memory address for the Nth memory operation.

58. The method of claim 56, wherein:

the first and second memory pointer registers provide memory addresses for store operations.

59. The method of claim 56, wherein:

the first and second memory pointer registers provide memory addresses for load instructions.

60. The method of claim 58, further comprising:

incrementing the values of the first and second memory pointer registers by M*x, where x is the size of the data to be stored during each of the store operations.

61. The method of claim 59, further comprising:

decrementing the values of the first and second memory pointer registers by M*x, where x is the size of the data to be loaded during each of the load operations.